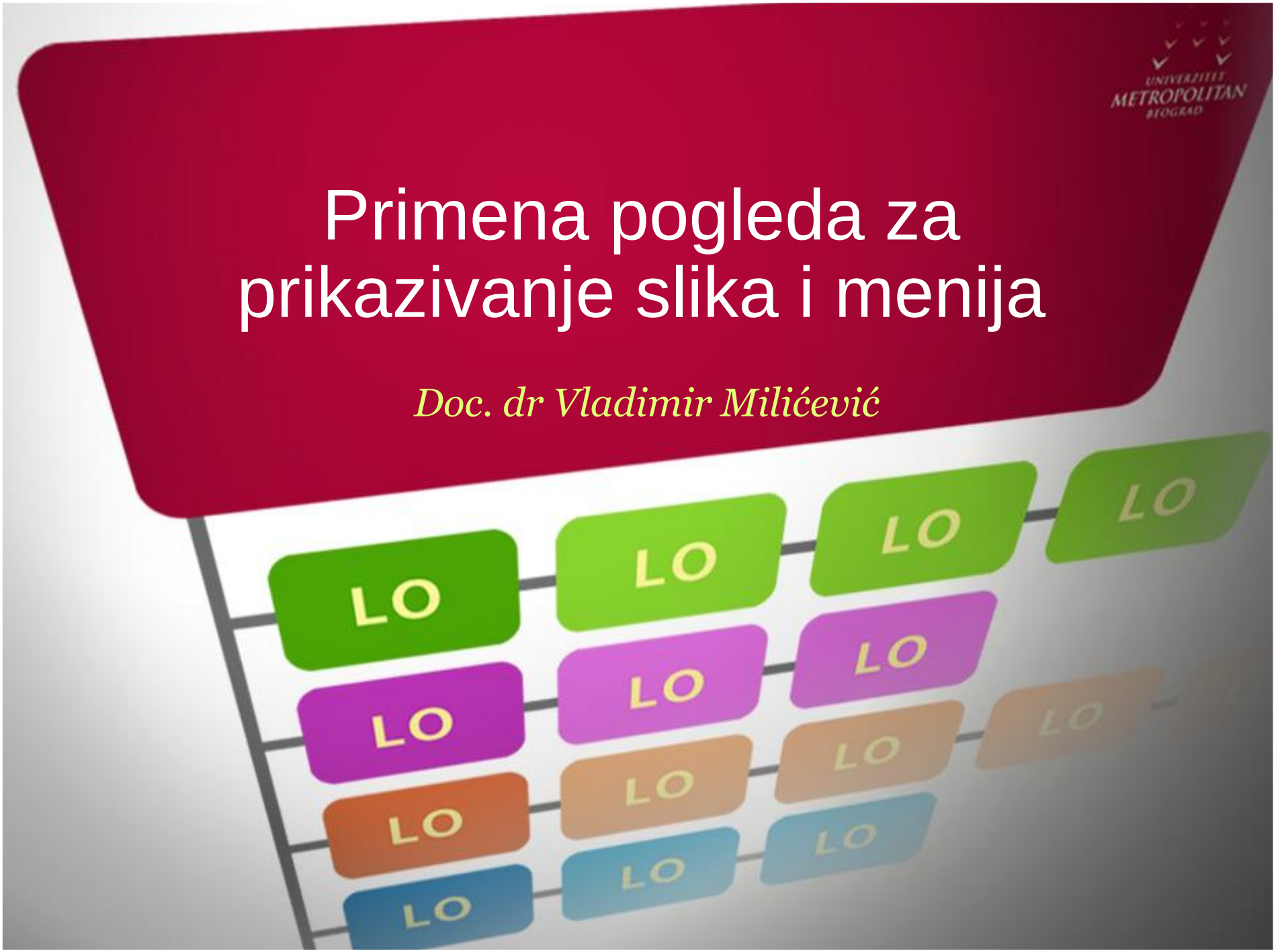
Primena pogleda za prikazivanje slika i menija

Doc. dr Vladimir Milićević

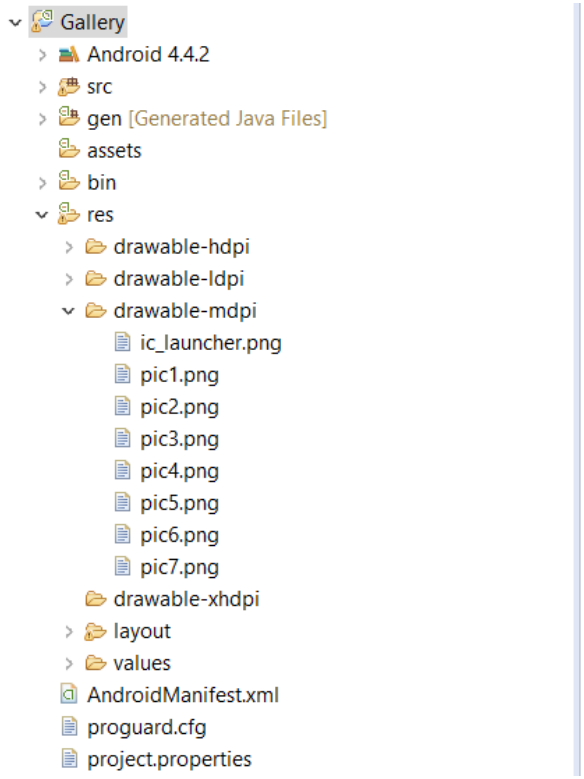


GALLERY I IMAGEVIEW POGLEDI

Gallery pogled predstavlja slike u formi horizontalne skrol liste, a ImageView prikazuje sliku koju je izabrao krosnik.

Gallery predstavlja horizontalnu skrolujuću listu čije elemente predstavljaju slike. Ako iz liste korisnik selektuje neku sliku, ona će na ekranu biti prikazana pomoću pogleda *ImageView*. Da bi skup slika, uopšte, mogao da bude prikazan listom, on mora da bude sačuvan u posebnom folderu pod nazivom *res/drawable-*.* Džoker znak * može imati neku od sledećih vrednosi: *hdpi* (visoka rezolucija), *ldpi* (niska rezolucija), *mdpi* (srednja rezolucija) i *xhdpi* (ekstra visoka rezolucija). U konkretnom slučaju slike će biti sačuvane u folderu *res/drawable-mdpi*.

Hijerarhija foldera projekta, sa folderom u kojem su sačuvane slike, data je sledećom slikom.



Slika-1 Hijerarhija projekta

GALLERY I IMAGEVIEW POGLEDI – XML DATOTEKE

Pored glavne XML datoteke, za definiciju UI, biće upotrebljena još jedna za deklarisanje stilova.

Gallery i *ImageView* pogledi biće definisani, kao deo Android korisničkog interfejsa, pomoću datoteke *main.xml*. Kod kojim će, na ovom nivou, pogledi biti realizovani, prikazan je sledećom slikom.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Slike Univerziteta Metropolitan" />

<Gallery
    android:id="@+id/gallery1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />

<ImageView
    android:id="@+id/image1"
    android:layout_width="320dp"
    android:layout_height="250dp"
    android:scaleType="fitXY" />

</LinearLayout>
```

Slika-2 main.xml projekta

Još jednu, pomoćnu, XML datoteku moguće je uvesti u projekat, a sa ciljem definisanja stilova prikaza. Datoteka će biti kreirana u folderu *res/values*, pored standardne *strings.xml* datoteke, i dobiće naziv *attrs.xml*. Kod je dat sledećom slikom.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="Gallery1">
        <attr name="android:galleryItemBackground" />
    </declare-styleable>
</resources>
```

Slika-3 Datoteka attrs.xml projekta

JAVA KLASA AKTIVNOSTI ZA GALLERY I IMAGEVIEW POGLEDE

Funkcionalnost aplikacije biće realizovana JAVA klasom GalleryActivity.java.

Funkcionalnost aplikacije biće realizovana JAVA klasom *GalleryActivity.java*. Klasa će upravljati odgovarajućim pogledima sa prikazivanje liste slika, i pojedinačih, a koristiće i *Toast* klasu za davanje informacija na ekranu o izabranoj slici iz liste. Kod *GalleryActivity.java* klase prikazan je sledećom slikom.

```
package net.learn2develop.Gallery;
import android.app.Activity;
public class GalleryActivity extends Activity {
    //---slike za prikazovanje---
    Integer[] imageIDs = {
            R.drawable.pic1,
            R.drawable.pic2,
            R.drawable.pic3,
            R.drawable.pic4,
            R.drawable.pic5,
            R.drawable.pic6,
            R.drawable.pic7
    };
    /** Kreira se kada se kreira aktivnost. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Gallery gallery = (Gallery) findViewById(R.id.gallery1);
        gallery.setAdapter(new ImageAdapter(this));
        gallery.setOnItemClickListener(new OnItemClickListener()
        {
            public void onItemClick(AdapterView<?> parent, View v,
            int position, long id)
            {
                Toast.makeText(getBaseContext(),
                        "Slika " + (position + 1) + " je izabrana",
                        Toast.LENGTH_SHORT).show();
                //---prikazuje izabrane slike---
                ImageView imageView = (ImageView) findViewById(R.id.image1);
                imageView.setImageResource(imageIDs[position]);
            }
        });
    }
   public class ImageAdapter extends BaseAdapter
    {
        Context context;
        int itemBackground;
```

```
import android.app.Activity;
import android.content.Context;
import android.content.res.TypedArray;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.Toast;
```

```
        public ImageAdapter(Context c)
        {
            context = c;
            //---podešava stil---
            TypedArray a = obtainStyledAttributes(R.styleable.Gallery1);

            itemBackground = a.getResourceId(
                R.styleable.Gallery1_android_galleryItemBackground, 0);
            a.recycle();
        }
        //---vraća broj slika---
        public int getCount() {
            return imageIDs.length;
        }
        //---vraća stavku---
        public Object getItem(int position) {
            return position;
        }
         //---vraća ID of stavke---
        public long getItemId(int position) {
            return position;
        }
        //---vraća ImageView pogled---
        public View getView(int position, View convertView, ViewGroup parent) {
            ImageView imageView;
            if (convertView == null) {
                imageView = new ImageView(context);
                imageView.setImageResource(imageIDs[position]);
                imageView.setScaleType(ImageView.ScaleType.FIT_XY);
                imageView.setLayoutParams(new Gallery.LayoutParams(150, 120));
            } else {
                imageView = (ImageView) convertView;
            }
            imageView.setBackgroundResource(itemBackground);
            return imageView;
        }
    }
}
```

Slika-4 Klasa aktivnosti primera

GALLERY I IMAGEVIEW POGLEDI – FUNKCIONISANJE

Neophodno je kreirati ImageAdapter klasu koja povezuje Gallery pogled sa serijom ImageView pogleda.

Kada je projekat kreiran, prva aktivnost je bila dodavanje definicije *Gallery* i *ImageView* pogleda u main.xml datoteku, a to se može videti iz priloženog koda.

Lista slika, koja će biti prikazana, spakovana je u niz pod nazivom *imageID* (sledeći kod).

Zatim je bilo neophodno kreirati i *ImageAdapter* klasu, naslednicu *BaseAdapter* klase, koja povezuje *Gallery* pogled sa serijom *ImageView* pogleda. *BaseAdapter* klasa povezuje *AdapterView* pogled sa izvorom podataka koji se prikazuju (videti kod priložen datotekom *GalleryActivity.java*).

```
//---slike za prikazovanje---
Integer[] imageIDs = {
        R.drawable.pic1,
        R.drawable.pic2,
        R.drawable.pic3,
        R.drawable.pic4,
        R.drawable.pic5,
        R.drawable.pic6,
        R.drawable.pic7
};
```

Slika-5 Niz slika za pakovanje u listu

AdapterView pogledi su: *ListView, GridView, Spinner* i *Gallery*. Neke od naslednica *BaseAdapter* klase su: *ListAdapter*, *ArrayAdapter, CursorAdapter, SpinnerAdapter* itd.

Posebno, metoda *getView()* ove klase, vraća *View* na tačno određenoj poziciji. U ovom primeru biće vraćena slika, odnosno *ImageView* objekat. Slike su spakovane u niz, a to znači da se po obavljenom izboru određuje i pozicija slike koja se prikazuje *ImageView* pogledom (sledeći kod).

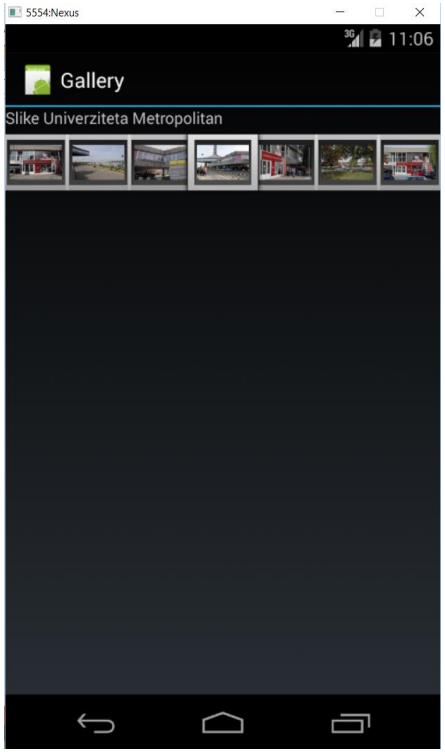
```
    Gallery gallery = (Gallery) findViewById(R.id.gallery1);
    gallery.setAdapter(new ImageAdapter(this));
    gallery.setOnItemClickListener(new OnItemClickListener()
    {
        public void onItemClick(AdapterView<?> parent, View v,
        int position, long id)
        {
            Toast.makeText(getBaseContext(),
                    "Slika " + (position + 1) + " je izabrana",
                    Toast.LENGTH_SHORT).show();
            //---prikazuje izabrane slike---
            ImageView imageView = (ImageView) findViewById(R.id.image1);
            imageView.setImageResource(imageIDs[position]);
        }
    });
}
```

Slika-6 Prikazivanje selektovanih slika

GALLERY I IMAGEVIEW POGLEDI – DEMONSTRACIJA

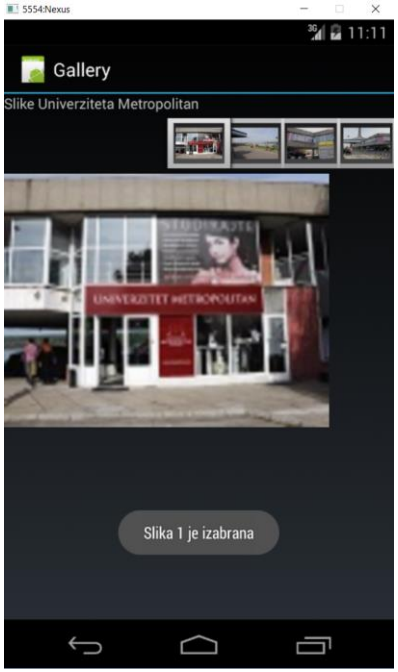
Prevođenjem i pokretanjem aplikacije na ekranu se pojavljuje interfejs sa dva pogleda.

Klikom na F11, aplikacija se prevodi i pokreće emulatorom. Početni ekran, pri čemu nijedna slika još nije selektovana, sadrži *Gallery* listu i prikazan je sledećom slikom.



Slika-7 Početni ekran aplikacije

Izborom neke od slika, na ekranu se pojavljuje i *ImageView* pogled koji prikazuje izabranu sliku. Takođe, klasom *Toast,* na ekranu će biti prikazan i odgovarajući komentar koji je u vezi sa izboranom slikom.



Slika-8 Izabrana slika iz liste

IMAGESWITCHER POGLED

Primena animacija na slikama moguća je kombinovanjem pogleda ImageSwitcher i Gallery.

U prethodnom izlaganju demonstrirana je saradnja pogleda *Gallery* i *ImageView*. Serija umanjenih slika je prikazana, a potom je jedna od njih izabrana i prikazana u većem formatu pomoću *ImageView* pogleda.

U nekim situacijama se ne traži brzo pojavljivanje neke slike već neka specifična akcija. Ta akcija može biti neka animacija koja se odnosi na prelazak sa jedne slike na drugu. Tada je neophodno koristiti *ImageSwitcher* pogled u kombinaciji sa pogledom *Gallery*.

Prethodni primer može da se modifikuje i da kvalitetno pokaže funkcionalnosti posmatranih pogleda.

Datoteka attrs.xml, iz foldera *res/value* će ostati nepromenjena, a isto će se desiti sa slikama koje su spakovane u folder *res/drawable-mdpi*.

Prva modifikacija biće učinjena main.xml datoteci zamenom XML atributa <ImageView> za <ImageSwitcher>. Navedeno je prikazano sledećim kodom.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Images of San Francisco" />

<Gallery
    android:id="@+id/gallery1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />

<ImageSwitcher
    android:id="@+id/switcher1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentBottom="true" />

</LinearLayout>
```

Slika-9 ImageSwitcher u main.xml

IMAGESWITCHER POGLED – MODIFIKACIJA JAVA DATOTEKE AKTIVNOSTI

Pored nasleđivanja klase Activity, klasa koja upravlja ImageSwitcher pogledom mora da implementira interfejs ViewFactory.

Klasa aktivnosti projekta, dobija na korišćenje još neke metode kroz implementaciju interfejsa *ViewFactory.* Kod klase aktivnosti i *ImageAdapter* klase nalazi se u istoj datoteci i prikazan je sledećom slikom.

```
package net.learn2develop.ImageSwitcher;
import android.app.Activity;
public class ImageSwitcherActivity extends Activity implements ViewFactory {
    Integer[] imageIDs = {R.drawable.pic1,R.drawable.pic2,R.drawable.pic3,
            R.drawable.pic4,R.drawable.pic5, R.drawable.pic6, R.drawable.pic7};
    private ImageSwitcher imageSwitcher;
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        imageSwitcher = (ImageSwitcher) findViewById(R.id.switcher1);
        imageSwitcher.setFactory(this);
        imageSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
                android.R.anim.slide_in_left));
        imageSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
                android.R.anim.slide_out_right));
        Gallery gallery = (Gallery) findViewById(R.id.gallery1);
        gallery.setAdapter(new ImageAdapter(this));
        gallery.setOnItemClickListener(new OnItemClickListener(){
            public void onItemClick(AdapterView<?> parent,
            View v, int position, long id)
            {
                imageSwitcher.setImageResource(imageIDs[position]);
            }
        });
    }
    public View makeView(){
        ImageView imageView = new ImageView(this);
        imageView.setBackgroundColor(0xFF000000);
        imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);
        imageView.setLayoutParams(new
                ImageSwitcher.LayoutParams(
                        LayoutParams.FILL_PARENT,
                        LayoutParams.FILL_PARENT));
        return imageView;
    }
```

```
public class ImageAdapter extends BaseAdapter{
    private Context context;
    private int itemBackground;

    public ImageAdapter(Context c){
        context = c;
        //---podešava stil---
        TypedArray a = obtainStyledAttributes(R.styleable.Gallery1);
        itemBackground = a.getResourceId(
                R.styleable.Gallery1_android_galleryItemBackground, 0);
        a.recycle();
    }
    //---vraća broj slika---
    public int getCount(){
        return imageIDs.length;
    }
    //---vraća stavku---
    public Object getItem(int position){
        return position;
    }
    //---vraća ID stavke---
    public long getItemId(int position){
        return position;
    }
    //---vraća ImageView pogled---
    public View getView(int position, View convertView, ViewGroup parent){
        ImageView imageView = new ImageView(context);
        imageView.setImageResource(imageIDs[position]);
        imageView.setScaleType(ImageView.ScaleType.FIT_XY);
        imageView.setLayoutParams(new Gallery.LayoutParams(150, 120));
        imageView.setBackgroundResource(itemBackground);
        return imageView;
    }
}
```

Slika-10 Klase aktivnosti i ImageAdapter

IMAGESWITCHER POGLED – NASLEĐIVANJE I PAKETI

Klasa koja rukuje ImageSwitcher pogledom koristi objekte i metode brojnih klasa.

Brojni paketi su uključeni u Android projekat koji rukuje ImageSwitcher pogledom. U ovim paketima nalaze se korisne klase čije objekte i metode koristi klasa aktivnosti aplikacije za postizanje konkretnih funkcionalnosti. Takođe, kroz nasleđivanje, klasa aktivnosti aplikacije nasleđuje funkcionalnosti bazne klase *Activity*. Sledećom slikom su prikazani paketi i njihove klase, angažovani u tekućem projektu.

```
*
import android.app.Activity;
import android.content.Context;
import android.content.res.TypedArray;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewGroup.LayoutParams;
import android.view.animation.AnimationUtils;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageSwitcher;
import android.widget.ImageView;
import android.widget.ViewSwitcher.ViewFactory;
```

Slika-11 Paketi i klase uključeni u klasu aktivnosti

IMAGESWITCHER POGLED – FUNKCIONISANJE

Metodom makeView() omogućena je upotreba ImageSwitcher pogleda.

Kao što je primećeno klasa aktivnosti aplikacije ne samo da je izvedena iz osnovne *Activity* klase, već i implementira *ViewFactory* interfejs. Ovaj interfejs definiše poglede koji se koriste uporedo sa *ImageSwitch* pogledom. Iz ovog razloga je neophodno implementirati *makeView()* metodu kojom je omogućeno umetanje novog pogleda u *ImageSwitch*, a taj pogled je, konkretno, *ImageView* (sledeća slika).

```
public View makeView(){
    ImageView imageView = new ImageView(this);
    imageView.setBackgroundColor(0xFF000000);
    imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);
    imageView.setLayoutParams(new
            ImageSwitcher.LayoutParams(
                    LayoutParams.FILL_PARENT,
                    LayoutParams.FILL_PARENT));
    return imageView;
}
```

Slika-12 Implementirana makeView() metoda

U metodi *onCreate()* učitava se *ImageSwitcher* objekat i podešava animacija. Na kodu, sa sledeće slike, vidi se animacija podešena konstantama *fade_in* i *fade_out*. Ovde je moguće dalje uvoditi novine. Ako se želi da se slika pojavi sa leve strane, a nestane sa desne, prilikom izbora naredne slike, moguće je koristiti konstante za animaciju: *slide_in_left i slide_out_right.*

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    imageSwitcher = (ImageSwitcher) findViewById(R.id.switcher1);
    imageSwitcher.setFactory(this);
    imageSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
            android.R.anim.slide_in_left));
    imageSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
            android.R.anim.slide_out_right));
    Gallery gallery = (Gallery) findViewById(R.id.gallery1);
    gallery.setAdapter(new ImageAdapter(this));
    gallery.setOnItemClickListener(new OnItemClickListener(){
        public void onItemClick(AdapterView<?> parent,
        View v, int position, long id)
        {
            imageSwitcher.setImageResource(imageIDs[position]);
        }
    });
}
```

Slika-13 Podešavanje animacije

GRIDVIEW POGLED

GridView pogled omogućava prikazivanje slika u matričnoj formi sa skrol opcijom.

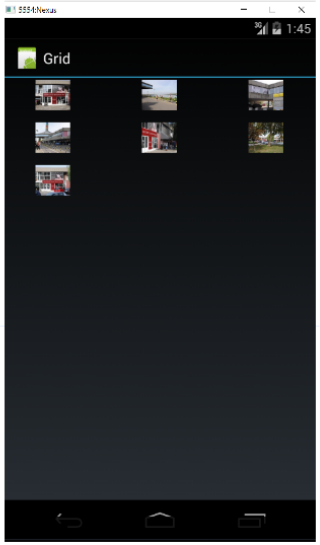
GridView pogled omogućava prikazivanje slika u dvodimenzionalnoj skrolujućoj mreži. Često se koristi u kombinaciji sa *ImageView* pogledom za prikazivanje serije slika u Android aplikacijama. Za demonstraciju moguće je uvesti izvesne korekcije u prethodni primer.

Kao i u prošlom slučaju, datoteka *attrs.xml* ostaće nepromenjena, ali će zato main.xml imati novu definiciju kao što je prikazano kodom sa sledeće slike. U sliku je ugrađena i reprezentacija korisničkog interfejsa na ekranu.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

<GridView
    android:id="@+id/gridview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
    android:horizontalSpacing="10dp"
    android:columnWidth="90dp"
    android:stretchMode="columnWidth"
    android:gravity="center" />

</LinearLayout>
```



Slika-14 Element <GridView> u main.xml

GRIDVIEW POGLED – JAVA KLASA AKTIVNOSTI PRIMERA

Klasa aktivnosti GridActivity je naslednica bazne klase Activity.

I u ovom slučaju, najveće modifikacije izvedene su nad JAVA klasama, klasi aktivnosti aplikacije, koja nosi naziv *GridActivity*, i klasi *ImageAdapter.* Ovde je već moguće konstatovati da je naučeno da klasa aktivnosti aplikacije nasleđuje baznu klasu *Activity*, a klasa *ImageAdapter* klasu *BaseAdapter*. Navedene klase su date kodom sa sledeće slike.

```
package net.learn2develop.Grid;
import android.app.Activity;
public class GridActivity extends Activity {
    //---Slike za prikazivanje---
    Integer[] imageIDs = {
            R.drawable.pic1,
            R.drawable.pic2,
            R.drawable.pic3,
            R.drawable.pic4,
            R.drawable.pic5,
            R.drawable.pic6,
            R.drawable.pic7
    };
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        GridView gridView = (GridView) findViewById(R.id.gridview);
        gridView.setAdapter(new ImageAdapter(this));

        gridView.setOnItemClickListener(new OnItemClickListener(){
            public void onItemClick(AdapterView<?> parent,
            View v, int position, long id){
                Toast.makeText(getBaseContext(),
                        "Slika " + (position + 1) + " je kliknuta",
                        Toast.LENGTH_SHORT).show();
            }
        });

    }

    public class ImageAdapter extends BaseAdapter{
        private Context context;

        public ImageAdapter(Context c){
            context = c;
```

```
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;
import android.widget.Toast;
```

```
    }

    //---vraća broj slika---
    public int getCount() {
        return imageIDs.length;
    }

    //---vraća stavku---
    public Object getItem(int position) {
        return position;
    }

    //---vraća ID stavke---
    public long getItemId(int position) {
        return position;
    }

    //---vraća ImageView pogled---
    public View getView(int position, View convertView,
    ViewGroup parent){
        ImageView imageView;
        if (convertView == null) {
            imageView = new ImageView(context);
            imageView.setLayoutParams(new
                GridView.LayoutParams(85, 85));
            imageView.setScaleType(
                ImageView.ScaleType.CENTER_CROP);
            imageView.setPadding(5, 5, 5, 5);
        } else {
            imageView = (ImageView) convertView;
        }
        imageView.setImageResource(imageIDs[position]);
        return imageView;
    }
  }
}
```

Slika-15 GridView pogled - JAVA klase

GRIDVIEW POGLED – FUNKCIONISANJE

Pomoću getView() metode moguće je odrediti veličinu slike i način na koji su slike međusobno razdvojene u GridView prikazu.

I u ovom slučaju, implementirana je *ImageAdapter* klasa koja je povezana sa *GridView* pogledom na sledeći način.

```
    gridView.setOnItemClickListener(new OnItemClickListener(){
        public void onItemClick(AdapterView<?> parent,
        View v, int position, long id){
            Toast.makeText(getBaseContext(),
                    "Slika " + (position + 1) + " je kliknuta",
                    Toast.LENGTH_SHORT).show();
        }
    });

}
```

Slika-16 ImageAdapter implementacija

Kada je obavljen izbor slike, *Toast* porukom šalje se informacija o selektovanoj slici na ekran uređaja. Pomoću *getView()* metode moguće je odrediti veličinu slike i način na koji su slike međusobno razdvojene u *GridView* prikazu izborom rastojanja za svaku pojedinačnu sliku (sledeći kod).

```
    //---vraća ImageView pogled---
    public View getView(int position, View convertView,
    ViewGroup parent){
        ImageView imageView;
        if (convertView == null) {
            imageView = new ImageView(context);
            imageView.setLayoutParams(new
                GridView.LayoutParams(85, 85));
            imageView.setScaleType(
                ImageView.ScaleType.CENTER_CROP);
            imageView.setPadding(5, 5, 5, 5);
        } else {
            imageView = (ImageView) convertView;
        }
        imageView.setImageResource(imageIDs[position]);
        return imageView;
    }
}
```

Slika-17 getView() metoda

MENIJI - PRIMENA POMOĆNIH METODA

Pre rada sa menijima, neophodno je kreirati dve pomoćne metode – za prikazivanje liste stavki i upravljanje događajima nakon izbora stavke menija.

Pre nego što programer kreira meni sa opcijama, ili kontekstni meni, prinuđen je da kreira dve pomoćne metode. Zadatak ovih metoda je prikazivanje liste stavki i upravljanje događajima nakon izbora stavke menija. Ove metode su deo klase aktivnosti projekta i u konkretnom slučaju biće nazvane *CreateMenu()* i *MenuChoice()*, a njihov kod je priložen slikom koja sledi.

```
private void CreateMenu(Menu menu){
    menu.setQwertyMode(true);
    MenuItem mnu1 = menu.add(0, 0, 0, "Stavka 1");{

        mnu1.setAlphabeticShortcut('a');
        mnu1.setIcon(R.drawable.ic_launcher);
    }
    MenuItem mnu2 = menu.add(0, 1, 1, "Stavka 2");{
        mnu2.setAlphabeticShortcut('b');
        mnu2.setIcon(R.drawable.ic_launcher);
    }
    MenuItem mnu3 = menu.add(0, 2, 2, "Stavka 3");{
        mnu3.setAlphabeticShortcut('c');
        mnu3.setIcon(R.drawable.ic_launcher);
    }
    MenuItem mnu4 = menu.add(0, 3, 3, "Stavka 4");{
        mnu4.setAlphabeticShortcut('d');
    }
    menu.add(0, 4, 4, "Stavka 5");
    menu.add(0, 5, 5, "Stavka 6");
    menu.add(0, 6, 6, "Stavka 7");
}
```

```
private boolean MenuChoice(MenuItem item){

    switch (item.getItemId()) {
    case 0:
        Toast.makeText(this, "Kliknuli ste na stavku  1",
            Toast.LENGTH_LONG).show();
        return true;
    case 1:
        Toast.makeText(this, "Kliknuli ste na stavku  2",
            Toast.LENGTH_LONG).show();
        return true;
    case 2:
        Toast.makeText(this, "Kliknuli ste na stavku  3",
            Toast.LENGTH_LONG).show();
        return true;
    case 3:
        Toast.makeText(this, "Kliknuli ste na stavku  4",
            Toast.LENGTH_LONG).show();
        return true;
    case 4:
        Toast.makeText(this, "Kliknuli ste na stavku  5",
            Toast.LENGTH_LONG).show();
        return true;
    case 5:
        Toast.makeText(this, "Kliknuli ste na stavku  6",
            Toast.LENGTH_LONG).show();
        return true;
    case 6:
        Toast.makeText(this, "Kliknuli ste na stavku  7",
            Toast.LENGTH_LONG).show();
        return true;
    }
    return false;
}
```

Slika-1 Pomoćne metode

POMOĆNE METODE - FUNKCIONISANJE

CreateMenu() preuzima argument tipa Menu, a MenuChoice argument tipa MenuItem.

Kreiranje pomoćnih metoda predstavlja polazni osnov za kreiranje aplikacije za rad sa menijima u Android operativnom sistemu.

CreateMenu() metoda preuzima argument tipa Menu i dodaje niz stavki menija. Za dodavanje stavke u meni, potrebno je kreirati objekat klase *MenuItem* i izvršiti metodu *add*() objekta *Menu* (pogledati priloženi kod metode). Metoda *add()* poseduje četiri argumenta:

1. *groupID* – identifikator grupe kojoj stavka menija pripada;
2. *itemId* – jedinstveni identifikator stavke grupe;
3. *order* – redosled u kome stavka treba da bude prikazana;
4. *title* – tekst koji se prikazuje za određenu stavku menija.

Metoda *setAlphabeticShortcut()* je iskorišćena za podešavanje prečice na tastaturi za izbor konkretne stavke manija. Metodom *setIcon()* definiše se slika koja se prikazuje kao određena stavka menija.

Metoda *MenuChoice()* preuzima *MenuItem* argument, proverava njegov Id i ispituje koja je stavka liste izabrana. U primeru je iskorišćena *Toast* poruka za prikazivanje rezultata izbora stavke na ekranu mobilnog uređaja

MENI SA OPCIJAMA

Funkcionisanje menija sa opcijama zahteva implementaciju izvesnih specifičnih metoda.

Da bi meni sa opcijama bio prikazan u izvesnoj aktivnosti, neophodno je implementirati metode *onCreateOptionsMenu()* i *onOptionsItemSelected()*. Prva metoda se izvršava kada korisnik klikne na taster *MENU*. U konkretnom primeru, pomoćna metoda *onCreateMenu()* izvršava se za prikazivanje menija sa opcijama. Nakon izbora, stavke iz menija, izvršava se metoda *onOptionsItemSelected()*, koja implementira metodu *menuChoice()* za prikazivanje izabrane stavke i sprovođenje predviđene akcije.

Za kreiranje aplikacija sa menijima, neophodno je uključivanje izvesnog broja paketa sa odgovarajućim klasama (sledeća slika).

```
import android.app.Activity;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuInfo;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
```

Slika-2 JAVA paketi i klase za rad sa menijima

Kod metoda za kreiranje i upravljanje menijem sa opcijama, dat je sledećom slikom.

```
package net.learn2develop.Menus;
import android.app.Activity;
public class MenusActivity extends Activity {
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btn = (Button) findViewById(R.id.button1);
        btn.setOnCreateContextMenuListener(this);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        CreateMenu(menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item){
         return MenuChoice(item);
    }
    private void CreateMenu(Menu menu){
       // kod metode je priložen
    }
    private boolean MenuChoice(MenuItem item){
         // kod metode je priložen
         }
     }
```

Slika-3 Korisne metode za opcione menije

ANALOGCLOCK I DIGITALCLOCK POGLEDI

AnalogClock i DigitalClock pogledi omogućavaju prikazivanje trenutnog vremena na ekranu.

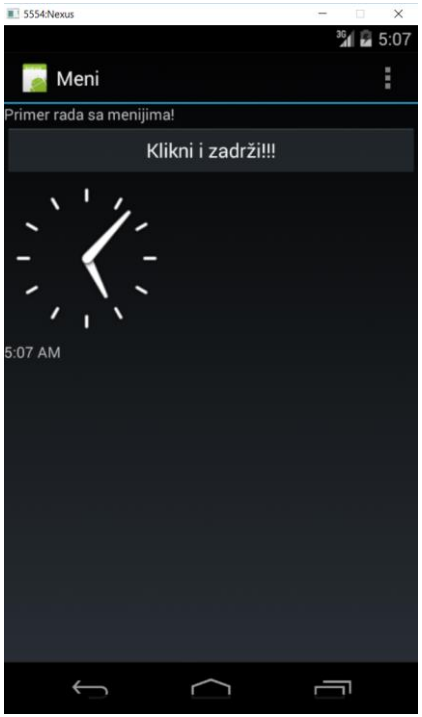
Android SDK, pored standardnih pogleda koji odgovaraju kontrolama korisničkog interfejsa, sadrži i dodatne poglede koji daju korisničkom interfejsu jedan lepši i bogatiji izgled. *AnalogClock* pogled omogućava prikazivanje trenutnog vremena, na ekranu mobilnog uređaja, u formi analognog časovnika sa kazaljkama za sate i minute. *DigitalClock* pogled prikazuje trenutno vreme u digitalnom formatu. Oba pogleda koriste sistemsko vreme preuzeto iz podešavanja mobilnog uređaja.

Za demonstraciju primera upotrebe ova dva pogleda, moguće je iskoristiti prethodnu aplikaciju za rad sa menijima. U main.xml datoteku biće dodata dva sledeća elementa.

```
<AnalogClock
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
     />
<DigitalClock
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    />
```

Slika-1 AnalogClock i DigitalClock - main.xml

Pozivom ctrl+s, biće snimljene promene u main.xml datoteci, a potom, klikom na F11, program se ponovo prevodi i pokreće. Rezultat pokretanja programa prikazan je sledećom slikom.



Slika-2 AnalogClock i DigitalClock - ekran aplikacije

UPOTREBA WEBVIEW POGLEDA

WebView pogled omogućava učitavanje web čitača u aktivnost.

Ukoliko Android aplikacija mora da ugradi specijalizovan sadržaj, poput web-a, u neku aktivnost, neophodno je da koristi poseban tip pogleda. Posebno koristan pogled, integrisan u Android SDK, jeste *WebView* pogled koji omogućava učitavanje web čitača u aktivnost.

Primena ovog pogleda biće demonstrirana sledećim primerom. U *Eclipse IDE* okruženju biće kreiran novi projekat sa nazivom *VebView*. Datoteka korisničkog interfejsa, main.xml, data je sledećim kodom.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

<WebView android:id="@+id/webview1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

</LinearLayout>
```

Slika-3 WebView primer - main.xml

Da bi aktivnost bila u mogućnosti da učitava web sadržaj, neophodno je u AndroidManifest.xml datoteku dodati sledeću privilegiju.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.WebView"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" />
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".WebViewActivity" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Slika-4 WebView primer - AndroidManifest.xml

UPOTREBA WEBVIEW POGLEDA – JAVA KLASA AKTIVNOSTI

WebView objektom vrši se učitavanje željenog url.

JAVA klasa aktivnosti implementira metode neophodne za učitavanje web stranica primenom *WebView* pogleda. Konkretno, takav zadatak može da se poveri metodi *loadUrl().* Za korišćenje kontrole zumiranja, potrebno je učitati osobinu *WebSettings* iz *WebView* pogleda, a zatim izvršiti metodu *setBuiltInZoomControls()*. Pre svega navedenog, neophodno je kreirati *WebView* objekat. Kod koji opisuje navedene akcije, dat je sledećom slikom.

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    WebView wv = (WebView) findViewById(R.id.webview1);

    WebSettings webSettings = wv.getSettings();
    webSettings.setBuiltInZoomControls(true);

    wv.loadUrl("https://www.facebook.com" +
     "/FakultetInformacionihTehnologijaUM");

    }
```

Slika-5 Metoda za učitavanje web sadržaja

Ponekad se dešava da prilikom učitavanja stranice, koja vrši preusmeravanje na drugu web stranicu, *WebView* pogled inicira pokretanje *Browser* aplikacije, ugrađene u Android uređaj, sa ciljem učitavanja web sadržaja. Da bi se to sprečilo i omogućilo učitavanje sadržaja u okviru tekuće aktivnosti i *WebWiew* pogleda, neophodno je implementirati *WebViewClient* klasu i predefinisati metodu *shouldOverrideUrlLoading()*. Navedeno je prikazano kodom sa sledeće slike.

```
private class Callback extends WebViewClient {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        return(false);
    }
}
```

Slika-6 CallBack klasa

UČITAVANJE WEB SADRŽAJA IZ DATOTEKE

Umesto http://... linka WebView može da prikaže sadržaj iniciran iz neke datoteke.

Ako postoji HTML datoteka, sačuvana u folderu *assets* projekta, ona može biti učitana u *WebView,* primenom metode *loadURL()*. Lokacija HTML datoteke, njen HTML kod i poziv iz klase aktivnosti, prikazani su sledećom slikom.

Slika-7 Učitavanje web-a iz fajla

HTML string je moguće i dinamički kreirati, u JAVA klasi aktivnosti projekta, a zatim ga učitati u *WebView* upotrebom metode *loadDataWithBaseUrl()*. Navedeno je prikazano sledećom slikom.

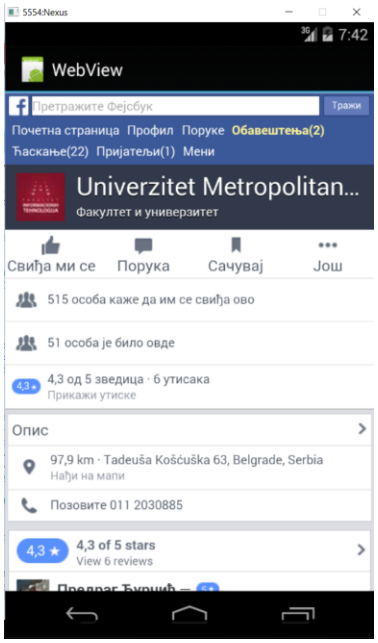
```
final String mimeType = "text/html";
final String encoding = "UTF-8";
String html = "<H1>Jednostavna HTML stranica</H1>" +
"<body>" + "<p>Prikaz malog web sadržaja!!!</p>" +
"<img src=\"http://master.rs/wp-content/uploads/2010" +
"/02/logo_bordo_pozadina-copy1.jpg\" />" +
 " </body>";
wv.loadDataWithBaseURL("", html, mimeType, encoding, "");
```

Slika-8 Dinami;ko učitavanje HTML-a

UPOTREBA WEBVIEW POGLEDA - DEMONSTRACIJA

Prevođenjem programa, za tri različita slučaja učitavanja sadržaja u WebView, vrši se demonstracija postignutih rezultata

Prvi slučaj, koji je razmatran, jeste učitavanje http://... linka metodom *loadURL()* iz klase aktivnosti projekta. Pozvani link odgovara *FaceBook* prezentaciji *Fakulteta informacionih tehnologija, Metropolitan Univerziteta.*



Slika-9 Učitavanje url-a u WebView

Sledeća dva slučaja podrazumevaju učitavanje HTML dokumenta u *WebView*. Prvi način je realizovan tako što je u folderu *assets* kreiran HTML dokument koji je učitan *loadURL()* metodom. U drugom slučaju, HTML dokument je predat kao string odgovarajućoj metodi u klasi aktivnosti projekta. Učitavanje HTML-a u *WebView* pogled prikazano je sledećom slikom.

Slika-10 Učitavanje HTML-a u WebView